Efficient Multi-viewpoint Acquisition of 3D Objects Undergoing Repetitive Motions

Yi Xu[*]
Purdue University

Daniel G. Aliaga[†]
Purdue University

Abstract

Computer graphics applications such as movie effects, video gaming, and product demonstration demand 3D models of dynamic objects. For this purpose, numerous methods, such as light fields, stereo reconstruction and visual hulls have been extended to model dynamic objects. These methods use multiple cameras to acquire images simultaneously and use the synchronized samples to reconstruct the model for each time instance. However, a large number of cameras are required to obtain compelling results. We introduce an efficient acquisition and modeling schema for dynamic objects with repetitive motions. Our method requires as few as two cameras. The key idea is that repetitive motions can be described by a finite number of states. Images capturing the same state can be grouped together and fed to the later modeling phase as if they are captured from multiple cameras simultaneously. Our work includes an acquisition system with interactive feedback, a graph traversal algorithm to help obtain a near minimum subset of images to sample the object and its motion, and a space-time image optimization method. We demonstrate this system using several datasets with different complexity of motion, and different number of desired viewpoints.

CR Categories: I.3 [Computer Graphics], I.3.3 [Picture/Image Generation], I.3.7 [Three-dimensional Graphics and Realism], I.4.1 [Digitization and Image Capture].

Keywords: computer graphics, image-based modeling and rendering, dynamic scenes, motion analysis, video textures.

1. Introduction

Obtaining models of dynamic 3D objects is an important part of content generation for virtual reality, movies, gaming, and several commercial applications. Modeling dynamic objects enables an observer to manipulate the object in both space and time. For example, a static observer can see the motion over time or a moving observer can freeze the motion and see the object along a viewpoint path. If the states (or poses) of the object repeat in the sequence periodically or quasi-periodically, we call such a *repetitive motion*. There are many rigid and non-rigid objects, such as toys, decorative items, and appliances, undergoing repetitive motions. In this paper, we are interested in efficiently capturing and modeling objects undergoing repetitive motions, rendering them from novel viewpoints, and generating new motion sequences.

Acquiring and modeling dynamic objects is a very challenging task. The process involves both creating a model of the object during each state of the motion and capturing the sequence of motion states. Motion capture methods focus on obtaining parameter values to control the shape and pose of a given object model during each state of the motion. Acquisition methods focus on creating an object model but not on identifying the repeating states of the motion. A process for both creating an object and capturing its motion can assume each state produces a unique projection onto the camera image plane [Allmen and Dyer 1990] and use a single static camera to determine the sequence of states. Unfortunately, simultaneously reconstructing the dynamic object for each state is difficult because it limits reconstruction to only the surfaces visible from a single viewpoint. Allowing the camera to move increases the visible surfaces, but complicates identifying the states of the repetitive motion and exacerbates the need for correspondence establishment unless the motion is perfectly periodic and the camera is capturing at a constant rate as in [Buehler et al. 2001].

Multi-camera acquisition methods can successfully identify the states of the repetitive motion and capture 3D objects by using a static installation of cameras surrounding the motion volume [Matusik et al. 2000; Zitnick et al. 2004; Starck et al. 2005; Vedula et al. 2005]. However, these methods have certain costs. First, they typically need a pre-installed and potentially-large calibrated infrastructure of synchronized cameras which is a stringent and expensive requirement. Second, the fixed installation often precludes portability and limits the flexibility to adapt to the number of images and viewpoints needed by the reconstruction algorithm. Third, mutual occlusions between cameras disallow certain configurations, such as one camera in front of another which is useful to obtain close-ups and higher-resolution views. In our work, we seek both to identify the state sequence of the repetitive motion and to reconstruct the 3D object, while overcoming the costs and limitations of large multi-camera acquisition systems.

Our key observation is that for repetitive motions we can combine the motion analysis benefits provided by a static camera with the object reconstruction ability of a moving camera observing a static scene. This allows us to model dynamic objects without the costs and limitations of large and static multi-camera acquisition systems and without having to tackle correspondence establishment of a moving camera seeing a moving scene. In particular, the static camera identifies a sequence of states which imitates the observed motion. The moving camera (or cameras), synchronized with the static camera, captures views of the object at a known state of its repetitive motion. Moreover, since the moving camera can be freely positioned, it can capture the same motion state from viewpoints in front of other camera viewpoints without generating an occlusion. In addition, by having knowledge of the motion sequence, the moving camera can obtain

[*]email: xu43@cs.purdue.edu
[†]email: aliaga@cs.purdue.edu

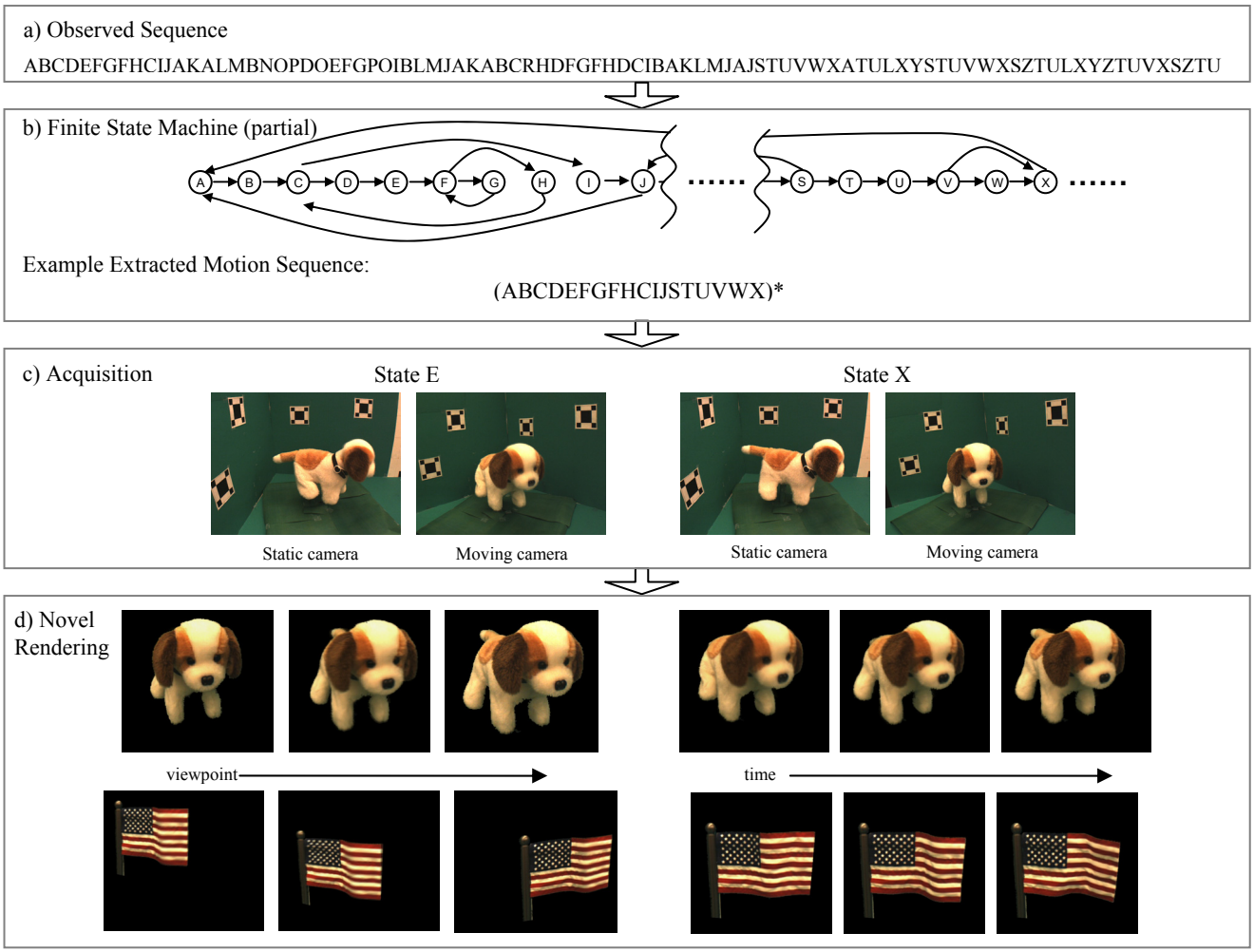

Figure 1. Capturing Objects undergoing Repetitive Motions. a) A static camera observes the dynamic object and identifies the unique states of the repetitive motion. b) Using the observed states and transitions, our algorithm constructs a finite state machine of the motion. c) Acquisition uses a static camera and at least one moving camera to efficiently capture all motion states from all desired viewpoints. d) Novel viewpoint renderings and motions are produced interactively for a moving observer (left) or for a static observer seeing the motion over time (right).

a complete set of images for any desired viewpoint density and camera speed. An acquisition that simply waves a camera around the object and ignores knowledge of the motion sequence might take significantly longer time or, in fact, never converge to a complete viewpoint sampling of all motion states. For a modeling method that requires few input images, such as image-based visual hull [Matusik et al. 2000], our approach can acquire a complete set of images quickly. If a denser set of image samples is required, as in light field [Gortler et al. 1996; Levoy and Hanrahan 1996; Buehler et al. 2001], our approach just captures a longer, but still compact, image sequence.

We present an efficient modeling and acquisition method for dynamic 3D objects undergoing repetitive motions (Figure 1). Our method uses one static camera and one or more moving cameras. In a brief preprocessing phase, the static camera observes the dynamic object and determines a sequence of *M states* of the repetitive motion which may be either periodic or quasi-periodic motion and may include rigid transformations and arbitrary object deformations, but the object must stay within a working volume. We create a graph with a node for each of *N* desired viewpoints of each state, resulting in a total of *N* x *M* nodes. Subsequently during real-time capture, the static camera classifies images to a state and the moving camera, which is synchronized to the static camera, captures each state from multiple viewpoints. To accelerate capture, we provide interactive feedback to guide the user to the right place at the right time and thus ensure desired sample coverage and reduce capture time. Once each state is sufficiently sampled, acquisition is complete. We then perform a silhouette-based volumetric reconstruction of each state of the acquired object and use a space-time optimization to align images over space and to stabilize the motion over time. The motion state sequence can be rearranged in order to produce new motion observable from novel viewpoints. We have captured several real-world objects undergoing a variety of repetitive motions. We demonstrate an efficient process, renderings from novel viewpoints, new motion sequences of captured objects, and multi-viewpoint captures not possible with a static arrangement of multiple cameras.

Our main contributions are as follows:

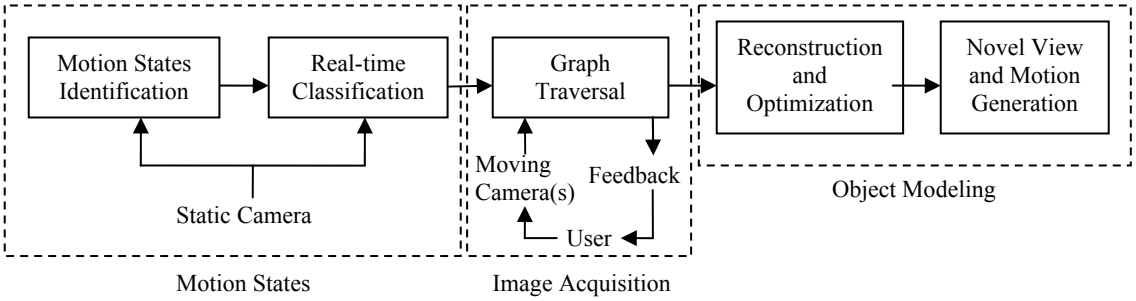

Figure 2. System Pipeline. Our approach consists of 3 major components: motion states, image acquisition, and object modeling.

- An approach using as few as two cameras to acquire and model objects undergoing repetitive motions thus enabling novel viewpoint rendering and new motion sequences.
- An acquisition graph and interactive feedback mechanism for ensuring desired image sample coverage and for reducing the capture time to acquire repetitive motion over multiple viewpoints and over time.
- An optimization method for aligning and registering images observing repetitive motion both spatially and temporally.

2. Related Work

Our research builds upon work in image-based object acquisition and repetitive motion analysis. While significant work exists in object acquisition, acquiring dynamic objects is still a major challenge for 3D geometry reconstruction [Hartley and Zisserman 2004] which often depends on establishing robust correspondences and/or limits the types of dynamic objects (e.g., articulated rigid objects [Yan and Pollefeys 2006]).

Image-based rendering (IBR) methods attempt to address some of the shortcomings of geometric reconstruction algorithms by directly re-sampling a large collection of source images in order to produce novel views of a captured object or scene, with no or only approximate geometry (e.g., [McMillan and Bishop 1995; Levoy and Hanrahan 1996; Gortler et al. 1996]). Capturing a dynamic object typically requires an array of (statically-calibrated) cameras. For example, J. Yang et al. [2002] uses 64 video cameras to capture a dynamic light field in real time. Naemura et al. [2002] use special hardware to estimate depth in real time and use a layered model to represent the scene. Wang et al. [2005] and Wilburn et al. [2005] use optical flow to guide spatial-temporal view interpolation. Other methods place the camera around the scene in order to acquire sufficient samples to reconstruct the scene geometry. Matusik et al. [2000, 2001] compute a representation for the visual hull directly from the background subtracted silhouette images. R. Yang et al. [2002] model the dynamic scene by projecting each reference image onto a series of planes that sweep the volume from back to front; and each pixel is shaded according to color consistency. Carranza et al. [2003] use silhouette images from multiple synchronized video cameras of a human actor to estimate the motion parameters and then to animate a human model for texture mapping. Zitnick et al. [2004] use region based stereo algorithm to reconstruct the dynamic scene and enables high quality free viewpoint video. All these methods have the high-cost of requiring (many) statically-installed cameras in order to obtain high quality results. The camera setup is also tuned to a particular modeling technique; for example, a planar array of cameras is good for capturing a light field but not good for reconstructing the visual hull of an object. Moreover, none of these approaches take advantage of the fact that the observed motion is repetitive.

Motion analysis work finds repeating patterns in an image sequence based on the assumption that 3D cyclic motion is preserved under single image projection. For example, Allmen and Dyer [1990] detect cyclic motion by finding repetitive patterns in the spatial-temporal curves that are defined on spatial-temporal surfaces caused by the motion. Cutler and Davis [2000] measure the self-similarity of a segmented object under repetitive motion and perform Time-Frequency analysis to characterize the periodicities. These two methods handle motion evolving in-place. Yet other methods handle global translation by alignment [Liu and Picard 1998; Laptev et al. 2005] or by tracking features [Seitz and Dyer 1997]. However, this kind of motion is not confined to a working volume and thus is not strictly repetitive. While these works analyze periodic motion, they do not focus on building a dynamic 3D model of the observed object over time.

Recently, a new type of medium called video textures has received research attention [Schödl et al. 2000]. A video texture is an infinitely varying image sequence that enhances images from a stationary viewpoint with projections of repetitive motions. Video textures have also been extended to large field of view panoramic images obtained by slowly rotating a planar camera around a stationary central axis [Agarwala et al. 2005]. In both cases, an input video clip is partitioned into states and smooth transition points in the video are found in order to produce a seemingly infinite loop of projected periodic and quasi-periodic scene motion but only from one fixed viewpoint.

In contrast, we seek multi-viewpoint acquisition of repetitive motions. We also wish to analyze the periodic motion and to take advantage of the recurrence of the motion states in order to improve the modeling of the observed dynamic object. Our approach is more general than perfectly periodic and/or human-motion specific methods. For example, Starck et al. [2005] use 10 cameras to capture multiple video sequences and use a motion graph [Kovar et al. 2002] tuned for predefined human motions. Similarly, Einarsson et al. [2006] present a method for capturing cyclic and constant speed human motions by using a turntable and a 1D camera array. Similar constant speed assumptions are used to create a motion lumigraph for a toy helicopter in [Buehler et al. 2001]. With the help of a static camera, our method robustly and during acquisition determines the states of arbitrary object motion.

3. Motion States

Our method partitions repetitive motion into a discrete number of states. The states are organized into a sequence that is representative of the observed repetitive motion. Then, in real

time, a captured image is classified as belonging to one of these states. Figure 2 contains a summary of our system pipeline.

3.1 Repetitive Motion

We categorize object motions into several different types based on the sequencing pattern of the states of the motion and we label all periodic and high-multiplicity random motion as repetitive motion. In particular, a static scene corresponds to a single state that repeats indefinitely. A completely random motion means every state is different and occurs only once. If some states appear more than once, then, based on the frequency of the appearance of the states, we define the motion as low multiplicity or high multiplicity random motion. If a whole subsequence repeats itself infinitely, the motion is periodic. Using the frequency of repetition of the states of the motion within a single period, periodic motion can also be sub-classified as low-multiplicity or high-multiplicity. In our work, we call periodic and high-multiplicity random motion repetitive motion. While periodic motion can be captured and reproduced perfectly, the acquisition of high-multiplicity random motion (or "quasi-periodic" motion) implies that a sufficiently similar motion sequence can be acquired. Our method is able to acquire objects undergoing such repetitive motion because each state will appear multiple times. For each appearance, acquisition strives to be *at the right place at the right time* so as to efficiently capture all states from a desired set of viewpoints in the viewing volume.

3.2 Identification and Classification

A single static, and uncalibrated, camera observes the dynamic object in order to identify the sequence of states of the repetitive motion as well as to classify a current image to one of the motion states. Using a short image sequence captured during a preprocessing phase, the motion is made to consist of one state for each captured image. Since we assume projections of the same state of the object motion are similar to each other, repetitions in the motion sequence are identified by measuring the image-difference between all pairs of captured images. Two states are merged into a single state only if the image difference between the representative images of the two states is below a predefined threshold. The threshold value controls the overall quality of the data fed to the modeling phase. A large threshold leads to a smaller number of states, but images of the same state can vary significantly. On the other hand, a small threshold leads to a larger number of states, but images within each state are very similar. The merging process repeats iteratively until no states are similar enough and a compact set of states is obtained.

Using the computed states and state transitions, the algorithm constructs a finite state machine that is representative of the observed repetitive motion. Although one would expect a perfectly repeating set of states for periodic motion, both periodic and quasi-periodic motion consist of a repeating motion sequence with spurious states appearing. In the case of periodic motion, the discrete time sampling of the camera may cause an image to be occasionally captured in between other motion states. For example, a pendulum can be observed from the static camera as predominantly having the state sequence $(ABCDCBA)^*$ but sometimes the sequence (ABCDCEA) may appear. A quasi-periodic motion is one that does not exactly repeat but exhibits significant similarities over time. The observed states and transitions that appear with frequency greater than a chosen threshold are placed into a state transition diagram, thus creating a finite state machine describing the major component of the motion. Our algorithm finds the largest strongly connected component of the state machine (thus ignoring dead-ends) and extracts those states as the representatives of the repetitive motion to be acquired (similar to the graph pruning in [Kovar et al. 2002]).

a) Observed state sequence

ABCDBCDBCEFGABCHFGABCEFGABCDBCEFGA

b) Computed finite state machine

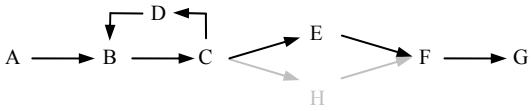

Figure 3. Motion States. a) Using the static camera, the observed repetitive motion is partitioned into states and state transitions. b) Both periodic and quasi-periodic motions are approximated by a finite state machine.

During live capture, our system matches the input image to cached images for each representative state in order to determine the current state of the repetitive motion. Our method performs an image-differencing operation and selects the best thresholded match as the current state. If an input image could not be matched to a state, it is discarded. This implies that the spurious motion states of the repetitive motion are appropriately ignored since they do not help to sample the desired motion.

Figure 3 shows an example process of identifying motion states and classifying input images. Given an initial input sequence of 34 frames, the system groups the motion into a high-multiplicity-periodic-motion sequence of only 7 states, and one spurious state which is discarded (state H in this case).

4. Image Acquisition

Given a desired viewing volume and viewpoint density, acquisition strives to sample all motion states from all desired viewpoints. A naïve approach might either randomly move the camera around the object and potentially never fully sample all motion states from all desired viewpoints; or perform a lengthy capture that places a movable camera at each desired viewpoint and captures an image sequence long enough to observe the entire repetitive motion sequence. However, due to the multiplicity of the motion states, this may not be optimal. For example, consider the motion of a pendulum or a fan: the left-to-right swing and right-to-left swing has many similar states. Instead of capturing an entire period at one viewpoint, a more efficient capture path will place the camera at one position, capture the left-to-right swing, then move the camera to the next position, capture the right-to-left swing. To generalize the notion to arbitrary repetitive motion sequences, potentially containing multiple repeating subsequences, we encode the state and viewpoint data into an acquisition graph and then the acquisition problem becomes how to efficiently traverse this graph.

4.1 Acquisition Graph

The images used for modeling the object and its motion are captured by at least one moving camera synchronized with the static camera. Since the static camera is able to determine the current motion state, the moving camera is, in fact, at all times aware of the current motion state. Thus, at every frame, the

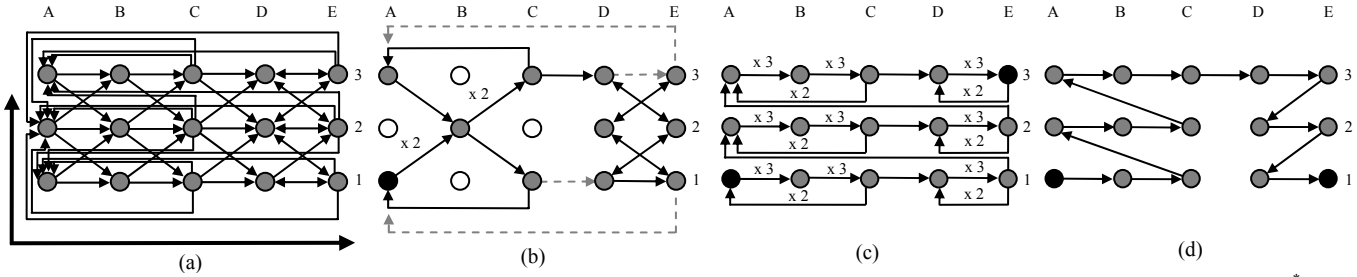

Figure 4. Example Graph Traversals. We show the graph and several traversals for the sequence (ABCABCABCDEDEDE)*. (a) The graph for 3 viewpoints (1-3), the 5 states (A-E), and all valid transition edges. (b) A simple zigzag pattern may lead to a very long capture sequence or, as in this example, never acquires all states from all viewpoints. (c) Staying at the same viewpoint until the entire motion repeats always yields a complete sampling but takes potentially a long time (45 time units in this example). (d) On the other hand, an ideal traversal completes in only 15 time units in this example.

moving camera is capturing a view of a known motion state from a potentially different viewpoint.

We place the acquired images into the nodes of a graph. For each of N desired viewpoints and for each of M motion states, there is a node in this graph, which results in N x M nodes in total. We label each node (v, s) corresponding to a viewpoint v observing a state s. Edges in this graph represent a possible movement from one desired viewpoint to another within an interval of time t. During time t (which is typically chosen to be one frame time), the object changes from motion state s_a to s_b. If the camera is fast enough to travel from a viewpoint v_1 to a viewpoint v_2, there is an edge from node (v_1, s_a) to node (v_2, s_b). A longer time interval t or a faster camera produces more edges between a node and its neighbors.

Figure 4a shows an example acquisition graph for the motion sequence (ABCABCABCDEDEDE)*. It contains all the nodes for five states and three viewpoints and all the valid edges assuming the spacing between desired viewpoints is equal to what the camera can move in one frame time.

4.2 Graph Traversal

The acquisition goal is to sample the object and its motion by efficiently visiting all nodes in the graph. Using a moving (handheld) camera to acquire images may result in many different types of graph traversals. Figure 4b-d show that various types of camera motion resulting in different traversals of an acquisition graph. Figure 4b depicts the traversal sequence for the intuitive motion of simply "zigzagging" the camera in front of the object (e.g., visiting viewpoints 1-2-3, then 3-2-1, etc. in this 1D example). Traversal starts at A-1, reaches E-3, and then the motion sequence restarts. Traversal continues from A-3 to E-1 and then goes back to A-1 and repeats indefinitely, leaving four nodes un-sampled. Although it is dependent on the viewpoint distribution, number of states, camera velocity, and motion sequence, such a simple unplanned motion might take a long time to capture all desired images or, in fact, not ever visit all nodes. Figure 4c represents a moving camera waiting for the entire repeating sequence to complete at each viewpoint. This approach always completes but might take a long time. On the other hand, Figure 4d shows the results of a moving camera following an ideal traversal sequence which yields a full sampling in potentially much less time.

4.3 Interactive Feedback

To improve live capture, we guide the moving camera to the right place at the right time. This guidance can serve to control a mechanical arm or as interactive feedback provided to the user holding a handheld camera. Our system provides immediate visual feedback to the user on the computer screen. The feedback system continuously estimates the camera's position, orientation, and velocity using a mechanically tracked arm attached to the moving camera (e.g., a MicroScribe G2LX arm). It also estimates the state of the motion in real time. According to the current camera viewpoint and the motion state, the system marks the corresponding node in the acquisition graph as sampled. In order to fully sample the motion, the user aligns the camera with one desired viewpoint and waits until all the states are sampled. Then the system signals the user to move to the next desired viewpoint.

Our feedback system can also guide the user to sample a subset of images very quickly when a coarse preview of the reconstructed object and its motion is desired. We use a heuristic search method to find the best place and direction for the user to move the camera in order to visit the un-sampled nodes in the graph as soon as possible. Starting at the node in the graph closest to the current viewpoint and given the movement direction, a recursive look-ahead of several steps is used to choose the next best edges (and viewpoints) that should be visited. Edges to nearby and un-visited viewpoints of the graph are preferred. To allow a typical smooth handheld viewpoint path, only viewpoints within a maximum angular deviation from the current movement direction are considered. This also prunes the recursive look-ahead tree and improves computational performance. As time advances, visited nodes are marked as such, and the last newly visited node becomes the current node. Furthermore, the set of edges of the graph can be quickly updated based on changes to the camera's position and speed.

5. Object Modeling

Our approach supports capturing and modeling objects using images with any desired viewpoint density and permits rearranging the captured object motion to produce novel motion sequences. For example, light field methods require a large number of planar images and small spacing between cameras. Structure from motion techniques need fewer images but require robust correspondence. Visual hull methods require a relatively small number of views but these views should observe the object from very different viewing directions. In this work, we model the dynamic object using a volumetric reconstruction algorithm. We also perform a spatial-temporal optimization to improve the captured motion states and their rearrangement.

5.1 Volumetric Reconstruction

We use a silhouette-based volumetric reconstruction method to build a voxel-based model for each state [Slabaugh 2004]. A silhouette map is computed for each frame from the moving camera by subtracting a certain background color. Camera pose for each frame is estimated by tracking feature points from a set of markers placed around the object. Voxels are then projected to each frame and the volumetric model is reconstructed by rejecting voxels that project outside the silhouette images. To render the object from a new viewpoint, we blend the projective texture maps of nearby reference images. Our reconstruction takes about 2-5 minutes per state on 6x6x6 voxels.

5.2 Spatial-Temporal Optimization

To improve the reconstruction, we perform an optimization of the camera frame poses within each motion state and among all motion states (Figure 5). The inaccuracies in feature tracking and calibration process as well as the discrete number of motion states lead either to visual popping during rendering when switching textures or to unnecessary image misalignment during a reconstructed motion sequence. To alleviate this, we reduce the re-projection error amongst images of the same motion state and improve the consistency of the object projection for an approximately stationary viewpoint over time.

Our optimization process minimizes the re-projection error using a sequence of key-frames. The motion consistency of the object is improved by choosing as key frames the sequence of sampled images over time (motion states) whose center-of-projection moves the least. Then, starting with the second motion state, the re-projection error of its key frame image to the key frame of the previous motion state is successively minimized by changing the pose of the most recent key frame. The re-projection error of the non-key frames to their respective key frames per motion state is diminished by changing the non-key frame poses. The non-key frames closest to the key frame are optimized first. Images far from the key frame are optimized to a nearby already optimized non-key frame.

5.3 Motion Rearrangement

Our method allows us to rearrange the order of the motion states producing new motion sequences and repetitive motions. The state transition diagram contains all the potential states and transitions. Similar to video textures [Schödl et al. 2000], we produce new and different repetitive motions by walking through the finite state machine in different ways. However, since for each state we already have sampled images and a reconstructed model, the observer can freely move the virtual viewpoint during the new motion sequence. For example, if a motion is $(ABABABABCDCDCD)^*$, then we can change it to $(ABCD)^*$, $(BCDCDA)^*$, or to $(AB)^*$ and in all cases allow the observer to continuously change the viewpoint.

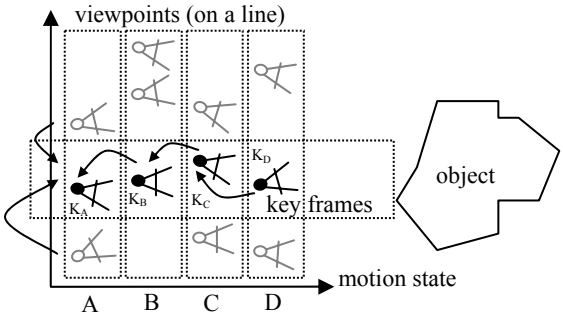

Figure 5. Optimization. Our method performs a space-time optimization: images within the same motion state are optimized to key-frames which correspond to images of sampled motions whose center-of-projection moves the least.

6. Results and Discussion

We have captured several objects using our approach and prototype implementation of one static camera and one moving camera. Our prototype system is implemented in C++ using standard OpenGL, GLUI, GLUT, and OpenCV libraries. The system uses two synchronized Point Grey Research (PGR) Flea cameras connected to a standard PC. The cameras capture color images of the object at 1024x768 pixel resolutions and at a frame rate of 15 Hz and are internally calibrated. State classification is done at the same frame rate using quarter-resolution images.

Table 1 lists four captured datasets, *puppy*, *flag*, *hamster*, and *fan*, each undergoing a different repetitive motion. For each dataset, we list the amount of time spent capturing images, the number of sampled states (on average, several spurious states are ignored per dataset), the number of captured viewpoints per state, the total number of stored images used for reconstruction, and the multiplicity of the motion. Since the motion does not need to be uniform, the multiplicity number only serves as an approximate measure of the number of times each state appears during an average cycle of the repetitive motion. We also list the final storage of the voxel models and textures in Table 1.

Figure 1 depicts the acquisition and reconstruction pipeline using the puppy and flag dataset. For the puppy, the static camera observes 24 states which re-appear in an approximately repeating sequence of 155 states (Figure 1a). From the observed states and transitions, our method automatically builds a finite state machine of the motion sequence (Figure 1b). During acquisition, the moving camera captures multiple views of each state and places them into a bin of images per state (Figure 1c). Our interactive feedback system guides the user through acquisition and ensures a complete and compact acquisition of the dynamic object and its motion. Only 624 images out of 6 minutes video (~5400 frames) are actually stored on disk for the puppy dataset. After object

Dataset	Capture Time	Valid States	No. Viewpoints	Images	Multiplicity	Storage (Voxel+Texture)
Puppy	~ 6 minutes	24	24	624	5.96	53M + 301M
Flag	~ 3 minutes	21	34	714	1	26M + 355M
Hamster	~ 3 minutes	10	29	290	11	52M + 126M
Fan	~ 4 minutes	86	12	1032	1.67	139M + 516M

Table 1. Datasets. We list the acquisition times and characteristics of the four datasets used for testing our method.

a) b) c) d)

Figure 6. Close-up Views and Optimization. (a) A view of the hamster only using images from a far. (b) Same viewpoint and time-instant of the motion but using close-up image that produces crisper, effectively higher-resolution, images. Unlike static multi-camera systems, we can place the moving camera in front of a previously captured viewpoint and obtain unoccluded images at multiple distances from the object. (c) Novel viewpoint rendering of flag without image optimization and (d) with optimization. Notice the reduced blurriness, especially noticeable around the stars.

reconstruction, a viewer can either freeze time and observe the object from a sequence of novel viewpoints (Figure 1d, left) or freeze the viewpoint at a new location and observe the motion over time (Figure 1d, right).

By efficiently traversing the acquisition graph, a user can sample an object and its motion with guaranteed completion and in relatively less time as compared to ad hoc moving-camera paths. Using the actual motion sequence of two datasets, we experimented in simulation with three camera speeds, three desired number of viewpoints, and four graph traversals. Table 2 shows three representative cases for each dataset. Using an incremental approach that remains at each desired viewpoint for an entire period may take significant time. Waving the camera in front of the object during its motion either takes a long time or never completely samples all states and viewpoints. In these examples, waving the camera captured only 70% of the images. On the other hand, our method outperforms the incremental approach and reaches 70% completion faster than zigzagging and always completed the sampling.

Since our approach does not require a large static installation of cameras, we can capture the motion using camera arrangements that would normally cause self-occlusions. For example, Figure 6b demonstrates a rendering from a novel viewpoint using captured images of the motion state obtained in front of and closer to the object as compared to the images used in Figure 6a for the same novel viewpoint. Thus, our method enables us to obtain close-ups of parts of the motion and produce seemingly higher-resolution imagery where and when desired.

Our space-time optimization process improves the rendering quality of the reconstructed objects and their motion. Figures 6c-d show the improved alignment between reference images before and after optimization. While the optimization process is automatic and does improve the quality, it comes at an additional computational cost of approximately 30-160 minutes per dataset.

Finally, the repetitive motions of any of our datasets can be altered yielding new and different motions. The user specifies a new way to traverse the state transition diagram and can still freely move the virtual viewpoint. Visually, this corresponds to rearranging the sequence of motion states observed in Figure 1d. More examples are in Figure 7 and in the paper video.

7. Conclusions and Future Work

We have introduced an efficient multi-viewpoint acquisition schema for dynamic objects undergoing repetitive motions. Our system uses only two cameras yet obtains samples of each motion state of a dynamic object from multiple viewpoints. By analyzing the motion in a pre-processing stage, we avoid storing a large number of redundant images and choosing a useful subset from them. By constructing an acquisition graph and providing feedback to the user, all motion states and desired viewpoints can be efficiently sampled. These samples are fed to a later modeling phase for reconstruction, optimization, and rendering of a 3D dynamic object. Furthermore, the captured repetitive motions can be altered and rearranged yielding new motions and still have viewpoint freedom. Our approach is also independent of the modeling technique in the sense that we can obtain images at any desired viewpoint density. Finally, our method also provides opportunities for capturing images from viewpoint configurations

Dataset	Camera Speed	No. Views	Incremental (100%)	Zigzag (70%)	Our Method (70%)	Our Method (100%)
Hamster	0.4	4x4	1968	555	210	1290
	0.2	8x8	7872	1140	720	3990
	0.1	16x16	31488	25155	3570	15930
Puppy	0.4	4x4	2752	1125	1080	1935
	0.2	8x8	11008	∞	4260	7800
	0.1	16x16	44032	∞	17520	31995

Table 2. Graph Traversals. We experimented in simulation with two datasets using different camera speeds, number of views, and traversal/motion methods. Camera speed is in terms of a viewing area of size one. Acquisition time is in units of per-frame time. Our traversals are consistently better.

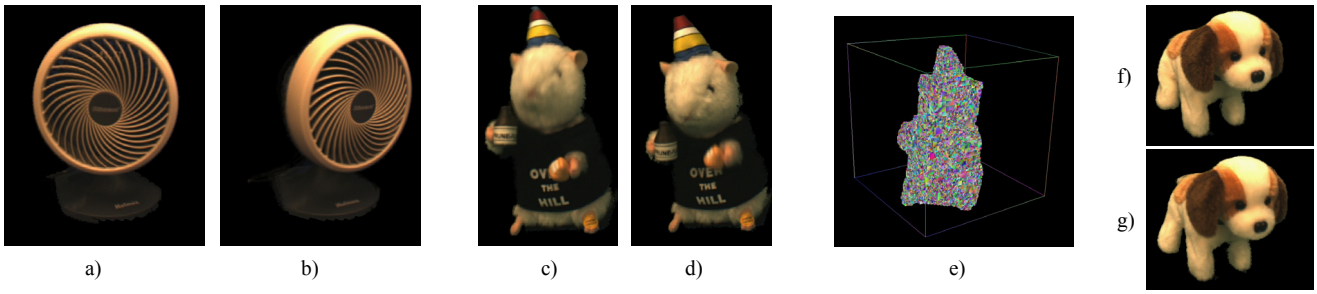

Figure 7. Example Novel Views. (a-b) Two novel renderings of the fan dataset, (c-d) and of the hamster dataset. (e) Screen-shot of underlying 3D voxel model for hamster. (f-g) Two novel renderings of the puppy dataset while sitting (f) and while standing (g).

not feasible with static multiple camera setups.

Our system has several current limitations as well potential directions for future work. First, we assume moving objects remain in a compact working volume. We would like to investigate how to capture local repetitive motion but under global translation and rotation. Second, we would like to study the theoretical aspect of how many static views are enough to uniquely identify all repetitive motions. Third, we would like to implement our system with a robot arm, which is controlled by the feedback system. This will yield better reconstruction quality and 360 degree acquisition as compared to handheld cameras. Finally, in our approach the quality of view-dependent illumination effects depends on the density of image capture; we would like to extend our technique to include highly-specular and inter-reflective objects (e.g., a water fall, candles, etc.) so as to expand our range of supported object types.

Acknowledgement

We would like to thank the reviewers for their suggestions to improve this paper. This work was supported by NSF CCF 0434398 and by a Purdue Research Foundation grant.

References

ALLMEN, M. DYER, C.R., "Cyclic Motion Detection using Spatiotemporal Surfaces and Curves", *Int'l Conf. on Pattern Recognition*, pp. 365-370, 1990.

AGARWALA, A., ZHENG, K.C., PAL, C., AGRAWALA, M., COHEN, M., CURLESS, B., SALESIN, D., and SZELISKI, R., "Panoramic Video Textures", *ACM Transactions on Graphics*, 24, 3, 2005.

BUEHLER C., BOSSE, M., MCMILLAN, L., GORTLER, S., and COHEN, M., "Unstructured Lumigraph Rendering", *ACM SIGGRAPH*, pp. 425-432, 2001.

CARRANZA, J., THEOBALT, C., MAGNOR, M.A., and SEIDEL, H.-P., "Free-Viewpoint Video of Human Actors", *ACM Transactions on Graphics*, 22, 3, 2003.

CUTLER, R. and DAVIS, L., "Robust Real-Time Periodic Motion Detection, Analysis, and Applications", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22, 8,781-796, 2000.

EINARSSON, P., CHABERT, C.-F., JONES, A., MA, W.-C., LAMOND, B., HAWKINS, T., BOLAS, M., SYLWAN, S., and DEBEVEC, P., "Relighting Human Locomotion with Flowed Reflectance Fields", *Eurographics Workshop on Rendering*, 2006.

GORTLER, S.J., GRZESZCZUK, R., SZELISKI, R., and COHEN, M.F., "The Lumigraph", *ACM SIGGRAPH*, pp. 43-54, 1996.

HARTLEY, R., and ZISSERMAN, A., "Multiple View Geometry In Computer Vision", *Cambridge University Press*, 2004.

KOVAR, L., GLEICHER, M., and PIGHIN, F., "Motion Graphs", *ACM Transactions on Graphics*, 21, 3, 2002.

LAPTEV, I., BELONGIE, S.J., PEREZ, P., and WILLS, J., "Periodic Motion Detection and Segmentation via Approximate Sequence Alignment", *Int'l Conf. on Computer Vision,* pp. 816-823, 2005.

LEVOY, M. and HANRAHAN, P., "Light Field Rendering", *ACM SIGGRAPH,* pp. 31-42, 1996.

LIU, F. and PICARD, R.W., "Finding Periodicity in Space and Time", *Int'l Conf. on Computer Vision*, pp. 376-383, 1998.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S.J., and MCMILLAN, L., "Image-based Visual Hulls", *ACM SIGGRAPH*, pp. 369-374, 2000.

MATUSIK, W., BUEHLER, C., and MCMILLAN, L., "Polyhedral Visual Hulls for Real-time Rendering", *Eurographics Rendering Workshop,* 2001.

MCMILLAN, L. and BISHOP, G., "Plenoptic Modeling: An Image-Based Rendering System", *ACM SIGGRAPH*, pp. 39-46, 1995.

NAEMURA, T., NITTA, T., MIMURA, A., and HARASHIMA, H., "Real-Time Video-Based Modeling and Rendering of 3D Scenes", *IEEE Computer Graphics and Applications,* 22, 2, pp. 66-73, 2002.

SCHÖDL, A., SZELISKI, R., SALESIN, D.H., and ESSA, I., "Video Textures", *ACM SIGGRAPH*, pp. 489-498, 2000.

SEITZ, S.M. and DYER, C.R., "View-Invariant Analysis of Cyclic Motion", *Int'l Journal of Computer Vision*, 25, 3, pp. 231-251, 1997.

SLABAUGH, G., CULBERTSON, W., MALZBENDER, T., STEVENS, M., and SCHAFER, R., "Methods for Volumetric Reconstruction of Visual Scenes", *Int'l Journal of Computer Vision*, 57, 3, pp. 179-199, 2004.

STARCK, J., MILLER, G., and HILTON, A., "Video-based Character Animation", *Symp. on Computer Animation*, 2005.

VEDULA, S., BAKER, S., and KANADE, T., "Image-based Spatio-temporal Modeling and View Interpolation of Dynamic Events", *ACM Transactions on Graphics*, 24, 2, pp. 240-261, 2005.

WANG, H. and YANG, R., "Towards Space-Time Light Field Rendering", *ACM SIGGRAPH Symp. on Interactive 3D Computer Graphics*, pp. 125-132, 2005.

WILBURN, B., JOSHI, N., VAISH, V., TALVALA, E.-V., ANTUNEZ, E., BARTH, A., ADAMS, A., HOROWITZ, M., and LEVOY, M., "High Performance Imaging Using Large Camera Arrays", *ACM Transactions on Graphics*, 24, 3, 2005.

YAN, J. and POLLEFEYS, M., "Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects", *Computer Vision and Pattern Recognition*, pp. 712-719, 2006.

YANG, J.C., EVERETT, M., BUEHLER, C., and MCMILLAN, L., "A Real-Time Distributed Light Field Camera", *Eurographics Workshop on Rendering*, 2002.

YANG, R., WELCH, G., BISHOP, G., "Real-Time Consensus-Based Scene Reconstruction using Commodity Hardware", *Pacific Graphics*, 2002.

ZITNICK, C.L., KANG, S.B., UYTTENDAELE, M., WINDER, S., and SZELISKI, R., "High-Quality Video View Interpolation using a Layered Representation", *ACM Transactions on Graphics*, 23, 3, 2004.